A QUANTITATIVE COMPARISON OF
STATE OF THE ART
CIRCLE DETECTION ALGORITHMS
Master of Science Thesis

Gökhan ÇIPLAK

Eskişehir, 2016

A QUANTITATIVE COMPARISON OF STATE OF THE ART CIRCLE DETECTION ALGORITHMS

Gökhan ÇIPLAK

MASTER OF SCIENCE THESIS

Department of Computer Engineering
Supervisor : Assist. Prof. Dr. Sevcan YILMAZ GÜNDÜZ

Eskişehir
Anadolu University
Graduate School of Science
November, 2016

FINAL APPROVAL FOR THESIS

This thesis titled “A Quantitative Comparison of State of the Art Circle Detection Algorithms” has been prepared and submitted by Gökhan ÇIPLAK in partial fullfillment of the requirements in “Anadolu University Directive on Graduate Education and Examination” for the Degree of Master of Science in Computer Engineering Department has been examined and approved on 14/11/2016

Commitee Members		**Signature**
Member (Supervisor)	: Assist. Prof. Dr. Sevcan YILMAZ GÜNDÜZ	.....
Member	: Assoc. Prof. Dr. Cihan KALELİ	.....
Member	: Assist. Prof. Dr. Mehmet KOÇ	.....

.....

Director

Graduate School of Science

ABSTRACT

A QUANTITATIVE COMPARISON OF STATE OF THE ART CIRCLE DETECTION ALGORITHMS

Gökhan ÇIPLAK

Computer Engineering Department

Anadolu University, Graduate School of Science, November, 2016

Supervisor: Assist. Prof. Dr. Sevcan YILMAZ GÜNDÜZ

Detecting circular objects in digital images are crucial problem in common applications. Although several circle detection algorithms have been released in the literature, the algorithms utilize a small set of images to show effectiveness. This situation causes unfair comparison between algorithms. In this thesis, a dataset including 200 images with size 800x6000 and and human annotations are proposed. Images in dataset have circular objects chosen from several application areas. The collected dataset is named as Anadolu University Circle Detection Dataset and Benchmark (AUCDB200), and is carried out for quantitatively comparison of the state of art circle detection algorithms in precision-recall-Fscore metrics. In this thesis, a novel circle detection algorithm is also proposed with benefiting from circular arcs of recently proposed Orientation Transform (OT). The novel algorithm is named as OTCircles. The experimental results in the thesis show that proposed algorithm, OTCircles, presents the best performance for proposed AUCDB200 dataset with 0.92 Fscore. The another results demonstrates that the algorithm is more robust against to noise.

Keywords: Circle Detection, Orientation Transform (OT), EDCircles.

ÖZET

MODERN DAİRESEL BÖLGE TESPİT ALGORİTMALARININ
NİCEL OLARAK KARŞILAŞTIRILMASI

Gökhan ÇIPLAK
Bilgisayar Mühendisliği Anabilim Dalı
Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Kasım, 2016
Danışman: Yard. Doç. Dr. Sevcan YILMAZ GÜNDÜZ

Bir çok uygulama alanına sahip olması, dijital imgelerde daire tespitini önemli bir problem yapmaktadır. Literatürde çok sayıda daire tespit algoritması sunulmasına rağmen, her bir algoritma kendi etkinliğini göstermek için özel seçilmiş imgeler kullanmaktadır. Dolayısıyla bu durum farklı algoritmaların birbiriyle kıyaslanmasına olanak tanımamaktadır. Bu problemi çözmek için bu tez çalışmasında 800x600 boyutlarında, çeşitli uygulama alanlarından dairesel nesneler içeren, işaretlemiş 200 imgeden oluşan dataset sunulmuştur. Bu dataset "Anadolu University Circle Detection Dataset and Benchmark (AUCDB200)" olarak adlandırılır ve bilinen daire tespit algoritmalarının duyarlılık - anma - Fscore ölçütleriyle nicel karşılaştırılmasında kullanılır. Bu tezde ayrıca yakın zamanda yayınlanan Orientation Transform ile çıkartılan yayların kullanılmasıyla, yeni bir daire tespit algoritması sunulmuştur ve bu algoritma OTCircles olarak adlandırılır. Deneyler, 0.92 Fscore değeri ile OTCircles'ın AUCDB200 dataseti için en iyi sonucu verdiğini ve gürültüye karşı daha az hassasiyeti olduğunu göstermektedir.

Anahtar Sözcükler: Daire Tespit, Orientation Transform (OT), EDCircles.

17/11/2016

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Anadolu University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

.....

Gökhan ÇIPLAK

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ABBREVIATIONS

AUCDB200 : Anadolu University Circle Detection Dataset and Benchmark

CACD : Curvature Aided Circle Detection

CHT : Circular Hough Transform

HT : Hough Transform

EDPF : Edge Drawing Parameter Free

FN : False Negative

FP : False Positive

GT : Ground Truth

ITCiD : Isosceles Triangle Circle Detection

NFA : Number of False Alarm

OT : Orientation Transform

RHT : Randomized Hough Transform

TD : Technical Drawing

TDLSF : Top-Down Least Square Fitting

TP : True Positive

TS : Traffic Sign

1. INTRODUCTION

Circular object detection in digital images is important problem in image processing and computer vision [1, 2]. Circle is one of the elementary geometrical shapes in nature. In everyday life, many circular objects are observed under telescope or microscope, in satellite images, in projects as technical drawings, and in several situation also encountered. Since circular objects contain valuable information in found or applied area, the detection of circles has important role in numerous applications ranging from pupil and iris detection [3–5], analysis of industrial goods [6, 7], ball detection in soccer matches [8], circular traffic sign detection [9–12], cell colony counting in biology [13], circular coins counting [14], and many several application areas.

Iris and pupil are circular parts of the eyes and pupil is located center of iris. Because of the fact that eyes are essential members of the face with their consistent location, detection of eye has crucial importance in head posture recognition [4]. Furthermore, iris and pupil detection play role in human computer interaction applications based on eye tracking [3]. Circle detection algorithms have priority in this applications. Also, circle detection algorithms are employed in production or industry. For instance, an example about foil alignment and inspection system in the work [6] has utilized circle detection methods to detect circular marker, which aims to reduce rotational offset and determine foil position. Food industry has huge diversity production and the production process. Image processing methods help for quality control with high accuracy and consistency due to the using robotic vehicles in production process [7] and many products are circular shape such as biscuits, cakes, and pizzas. Improving in vehicle technology, several artificial intelligence systems such as drive assist and autonomous driving have placed in the technology. Detection of traffic sign is the one of the developing driver assistance system. Traffic sign detection and recognition is the frequently applied area for circle detection algorithms [9–12]. Using circle detection methods, traffic signs may be detected and recognized while driving. Thus, driver may be informed about the detected traffic sign. Because morphology of some biological cells is similar to circular shape, circle detection methods have also been applied to cell colony images for counting amount of cells. In the work [13], cell colony images are handled to automatically count and analyze. Circe detection algorithm is employed to the image after preprocessing steps. Whereas coin detection and recognition problem have been solved using mechanical solutions, image processing solutions have also been proposed in literature [14]. Geometrical properties of

coins have been used for counting them in the proposed work. The core step in this work is detection of coin boundaries, which are circular shape.

Because circle detection is crucial problem and an area of interest for researchers, several circle detection algorithms have been proposed in the literature. Hough Transform (HT) [15, 16] is a generic method to detect lines, circles, ellipses or similar such shapes using a voting process and its parameter space, but requires high memory and computation. Circular Hough Transform (CHT) [16–19] detects circles in 3D parameter space, which contains the coordinates of the center pixel and the radius. The circles defined as right cones intersecting at a single point in the parameter space. However, the method demands high memory and has high execution time. To overcome the problems of CHT, various modification versions have been applied and proposed several methods including probabilistic HT [20–25], fuzzy HT [26], randomized HT (RHT) [27, 28], etc. Probabilistic approach is based on maximum likelihood parameter estimation using probability density function to determine unknown parameters [21]. Fuzzy approach approximately determines the position of the shape and while it is better than conventional HT in noise images, it needs membership function which is determined with experimentally [26]. RHT [27] randomly picks n pixels and then maps pixels to a point in the parameter space. This method detects curves and has lower memory demand than classical HT. Researchers in [29] also propose a RHT based line and circle detection algorithm to increase accuracy and robustness. It carries out detection with using two edge pixels that have close tangent value and one randomly picked pixel. Various kind of RHT methods have been utilized for circle detection [30, 31].

Besides applications of HT methods on circle detection, various randomized methods have been proposed to detect circles. Researchers in [32] have proposed a Randomized Circle Detection (RCD) algorithm. The algorithm randomly picks four edge pixels and generates four candidate circles from the combination of three non-collinear points. Circle is detected when the pixel not used while circle generation is in a specified distance from the circle. Once the distance is in specified distance, four edge pixels are defined as circular. To enhance execution time of this RCD, Chung & Huang [33, 34] have proposed efficient sampling and refinement strategies and use a lookup table based voting algorithm. According to the LUT-based platform, points in edge map are carried out the query on look up table, which is 2D binary array that provides to check whether given edge pixel on candidate feature, for voting process. Multiple evidence sampling strategy

involves three evidence. First one discards possible circles due to the its gradient direction. Second evidence checks fourth edge pixel distance and the last evidence controls gradient directions of four edge pixels to the center of the circle.

Genetic algorithms have also been applied to the circle detection problem [35–41]. Yao et al. [35] have proposed a ellipse detection algorithm using genetic algorithms. In the proposed work, not only evolution is carried out but also clustering of detected ellipses are handled. It also produces more accurate results than RHT. Ayala & Ramirez et al. [36] develop a genetic algorithm to detect circles. The work uses three edge points as chromosome. Result of the evolution is the detected circles. However, the algorithm has only difficulty imperfect and small circles in the image. Das et. al [37] have proposed a hybrid method consisting of simulated annealing and differential evolution. Simulated annealing, which is a heuristic search algorithm, is employed as selection method of differential evolution with a fuzzy fitness function. Dasgupta et al. [38] have proposed circle detection algorithm using adaptive bacterial foraging algorithm based on the swarm intelligence technique. The algorithm produces satisfactory detection results. However, it is sensitive to noise. Cuevas has proposed several circle detection algorithms with using several optimization techniques including discrete differential evolution [39], artificial immune systems [40], and harmony search optimization [41]. The discrete differential evolution algorithm takes three non-collinear edge points and its combination to decide candidate circles and is applied to edge map for circles in the image. Candidate circles are evolved utilizing discrete differential evolution to detect circles. The artificial immune systems method evolves the candidate circles using clonal selection algorithm. In harmony search optimization, the candidate circles, which are generated with encoding of three non-collinear edge points, have been evolved with harmony search algorithm. Though, the optimization techniques mentioned above produce satisfactory results, they have high execution time.

Apart from the genetic algorithms based circle detection, various real-time circle detection algorithms have also been developed. Zhang & Liu [42] propose a real time ellipse detection algorithm for face detection problem. Since ellipses are presented 5D accumulator in HT, the proposed method decompose the parameter space to obtain computationally efficiency. Frosio et al. [43] have presented a real time circle detection algorithm based on maximum likelihood. However, their algorithm needs apriori information about the radius of the circular objects to be detected. This predefined information

restricts the usability of the method for real time applications. In [44], the authors have proposed a high-speed circle detection algorithm; but experimental results is not enough to recognize their claim. The algorithm clusters connected or near edge pixels due to the meeting criteria. Then, radius and center of the circle which is generated by clustered pixels is calculated. Experiments of the algorithm has been carried out on 174 circles with resolution 640x480. Akinlar & Topal [45] have proposed a real time circle detection algorithm with a false detection control based using the Helmholtz principle [46–49], named EDCircles. The algorithm benefits from Edge Drawing Parameter Free (EDPF) [50] algorithm while extracting edge segments. Arcs are produced with combining small line segments and candidate circles generated with the arcs. Validated circles are the output of the algorithm. While the algorithm has high speed and precision, it has recall problem due to the arc segment extraction.

Recently, several studies have been presented for circle detection in the literature. Liu et al. [51] have proposed a circle detection algorithm based on a top-down scheme. This scheme finds circular arcs on sub-chains of the edge segment from longer arcs to shorter arcs. The similarity of this work with EDCircles [45] is validation process, which the process is carried out same as in EDCircles. Marco et al. [52] have proposed a randomized circle detection algorithm based on the analysis of the curvature of isophotes. This means that curves connecting pixels have same intensity value. Curvature of the isophotes are utilized to divide edge pixels into same curvature clusters. Zhang et al. [53] have also proposed a randomized circle detection algorithm. The algorithm utilizes isosceles triangles sampling strategy with gradient information of edge pixels without classical sampling strategies. Yao & Yi [54] determines circles in digital images using curvature aided Hough transform. This method finds circles with approximating circle radius and it's center from curvatures.

As summarized above, though there is numerous circle detection algorithms found in the literature, each algorithm utilizes a small set of specially chosen images. The aim of using this small set of images is to show effectiveness of the presented algorithm. This situation makes the algorithm robust and causes quantitative comparison of the performance of different algorithms impossible. To overcome this situation, in this thesis, a dataset consisting of 200 images is proposed, which each of images has size 800x600 and they are collected from different web sources [59–63] and images containing circular objects from a wide variety of application areas. Also, dataset contains with human

annotated ground truth. The proposed dataset is released under the name Anadolu University Circle Detection Dataset and Benchmark (AUCDB200) [64]. This dataset is utilized to quantitatively compare some of the state of the art circle detection algorithms in the precision-recall-Fscore metrics. The another contribution of this thesis is a new circle detection algorithm. This new algorithm, OTCircles, benefits circular arcs obtained from the recently proposed Orientation Transform (OT) [55]. The proposed circle detection benchmark is carried out to quantitatively compare various recently proposed circle detection algorithms with each other and also with OTCircles in precision-recall-Fscore metrics. Result of this work is that OTCircles shows the best performance for AUCDB200 with 0.92 overall Fscore and is less sensitive to Gaussian noise.

The organization of this thesis is as follows: Chapter 2 consists of a description of the collected dataset and the circle annotation tool develop to mark the ground truth circles. Chapter 3 gives the details of some of the recently published circle detection methods that are used in the quantitative comparison. Chapter 4 describes the proposed circle detection algorithm, OTCircles. Chapter 5 describes our quantitative evaluation methodology. Experimental results in given in chapter 6, and the thesis is concluded in chapter 7.

2. DATASET & ANNOTATION TOOL

Circle detection problem has been studied for a long time and many solutions have been proposed as mentioned in chapter 1. However, the performance of the proposed algorithms are demonstrated using a limited number of images without any ground truth. This situation causes unfair comparison between algorithms. To overcome this problem, a standard dataset that can be used for quantitative evaluation of the circle detection algorithms is prepared. Section 2.1 describes the details of the dataset, and section 2.2 describes the circle annotation tool.

2.1. Dataset

Circle detection is an important task in image processing and computer vision. Since circle is one of the basic shapes in nature and encompasses valuable information, circle detection is important and has many applications.

Several circle detection algorithms have been proposed in literature; but this area has still kept attraction by researchers. However, the shortage of the dataset used in literature causes to not enable to quantitatively and objectively comparison of various algorithms' performance. 200 color images involving circular items from a broad a range of implementation areas have been collected to overcome shortage in dataset. The images in the collected dataset have size 800x600 pixels and are in bitmap format. The images in the collected dataset are grouped into six groups and listed in Table 2.1. The groups consist of Balls, Circles, Satellite, Structure, Technical Drawing (TD), and Traffic Sign (TS) groups and each part contains 50, 25, 25, 25, 25, and 50 images respectively.

Table 2.1. *Categories, the number of images in each category, and the total number of circles within the images in each category of AUCDB200.*

Category	**Number of Images**	**Number of Circles**
Balls	50	729
Circles	25	688
Satellite	25	119
Structure	25	59
Technical Drawing (TD)	25	191
Traffic Sign (TS)	50	113
Total	**200**	**1899**

The characterization of these groups is listed as followings:

Balls (50 images) This group largely involves spherical objects like ball such as tennis, golf, billiard beads. (Figure 2.1(a))

Circles (25 images) This group largely involves circular items 2D plane. (Figure 2.1(b))

Satellite (25 images) This group largely involves circular items captured from Google Earth application, contains largely circular crossroads and oil tanks. (Figure 2.1(c))

Structure (25 images) This group largely involves huge circular items like meteorology stations. (Figure 2.1(d))

Technical Drawing (TD) (25 images) This group largely involves hand and computer circle drawing images with different thicknesses. (Figure 2.1(e))

Traffic Sign (TS) (50 images) This group largely involves circular traffic signs. (Figure 2.1(f))

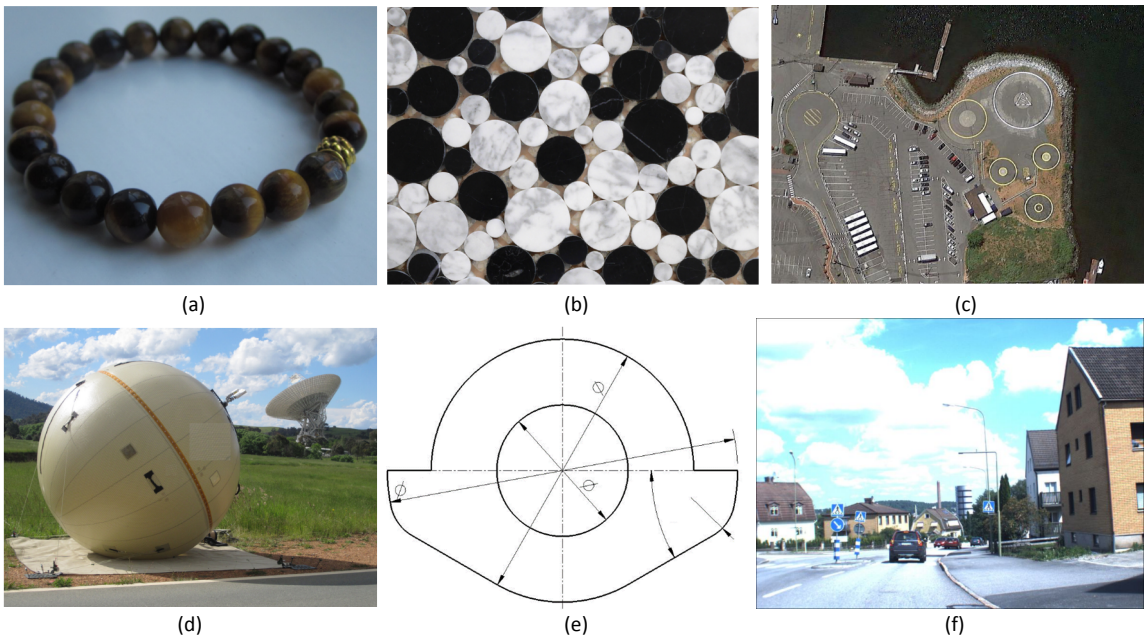
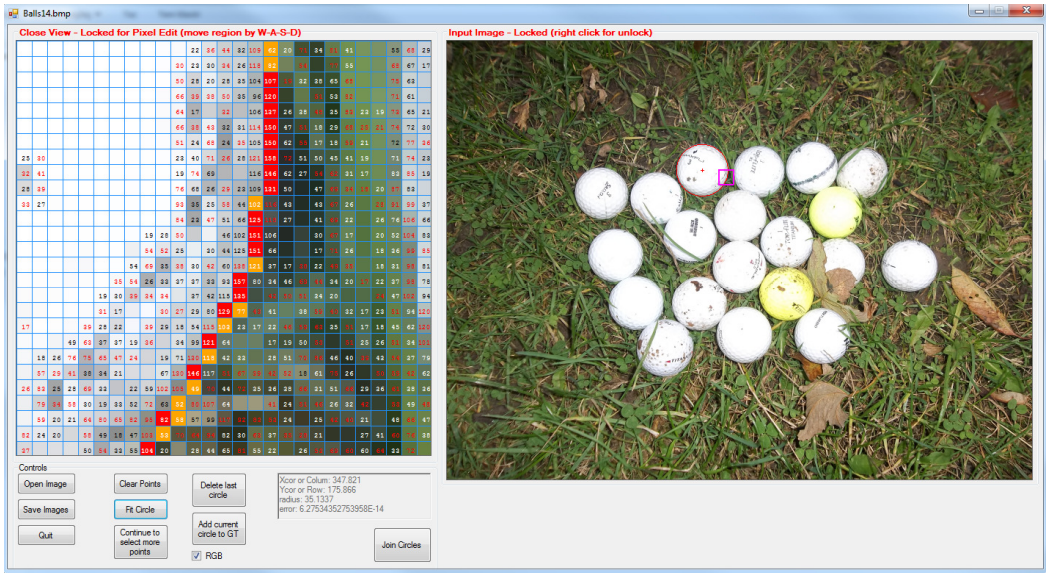


Figure 2.1. *(a) is an image from group Balls, (b) is an image from group Circles, (c) is an image from group Satellite, (d) is an image from group Structure, (e) is an image from group Technical Drawing, (f) is an image from group Traffic Sign*

2.2. Annotation Tool

In this thesis, a circle annotation tool has been developed. This tool enables users for easier modification and annotation of circles in the image within the dataset. Figure



(a)

(b) (c)

Figure 2.2. *(a) Main window of the annotation tool. The right side of the window shows the image being edited, with the left side showing a zoomed-in portion within a region of interest square, (b) the window showing the currently fitted circle, (c) the window showing the set of marked circles in the Ground Truth (GT).*

2.2 demonstrates a view of user interface (UI) of the developed tool. The tool has three windows as seen in Figure 2.2. The primary window shown in Figure 2.2(a) involves several functions to load images, annotate circles, and save the ground truth data to a text file. The image being annotated is placed at the right side of the primary window. The image that is placed left side of the primary window is an expanded part of the image marked with a region of interest square. This region may be determined by mouse movements on the image.

The working of the annotation tool is performed as following: The user picks at least three pixels over the contour of the considered circle. After, user clicks the "Fit Circle" button placed at the bottom-left corner of the primary window to annotate a circle on the

image. The generated circle is then appeared on the right side of the primary window and also on the second window demonstrated in Figure 2.2(b). Enhancement on the generated circle is carried out using "Continue to select more points" button. With this button, user picks more pixels on the contour and continues with pressing "Fit Circle" to generate new enhanced circle. This train may carry on only if the generated circle does not satisfy the user. When the user wants to start picking the pixels at beginning, the user may use "Clear Points" button. This button will clear all picked pixels by user. Whenever the user is pleased with the generated circle, user may click "Add current circle to GT" button to add this generated circle to the current the ground truth (GT) set of circles. Whole circles in the ground truth set are appeared in the last window demonstrated in Figure 2.2(c). The figure demonstrates annotated circles as an example.

The annotated circles are saved to a text. Each line of the text file indicates a circle with its properties x, y, and radius. Furthermore, the annotated circles demonstrated in Figure 2.2(c) are also saved to a bitmap file for easily perception and visualization by users. Also, when the image is reload, the tool loads not only it but also its ground truth named as combination of the image file name and "_GT" with ".txt" extension if it exists. Using this tool, whole GT circles in 200 color images of AUCDB200 have been annotated, and publicly released from [64]. It is expected that released dataset will gain acceptance by the concerned community and carry out to the quantitatively evaluate for any developed or being developed circle detection algorithms.

3. RECENTLY PUBLISHED CIRCLE DETECTION ALGORITHMS

In this chapter, the goal is to elaborate the details of four recently proposed circle detection algorithms, which being used for quantitative comparison with OTCircles. EDCircles [45], which forms the basis of OTCircles, is firstly described. Then Curvature Aided Circle Detection (CACD) [54] is reviewed. Next, Top-Down Least-Square Fitting analysis (TDLSF) for circle detection [51] is examined. Finally, Isosceles Triangle sampling based Circle Detection (ITCiD) [53] algorithm is reviewed.

3.1. EDCircles: A Real-Time Circle Detector With A False Detection Control

EDCircles [45] is a real time circle and ellipse detection algorithm. The main idea of the algorithm is based on the extraction lines and generating circular arcs from them. The algorithm has several steps.

Firstly, it detects edge segments with edge drawing parameter free (EDPF) [50], which runs edge drawing [56] with its parameters set to extreme values. Thus, it extracts all possible values with many false positive edge segments. These false positive segments are eliminated using the Helmholtz principle [46–49] and the algorithm leaves only meaningful edge segments. Some of these edge segments may be closed, i.e., the first and the last pixels of the segment are neighbors, and thus, they may directly trace the boundary of a circular object. These edge segments are fitted to circles using least squares circle fitting algorithm [65], and if the circle fit error is smaller than some threshold, then the circle is added to the candidate circle list. Such edge segments are then removed from further consideration.

Secondly, the remaining edge segments are converted to line segments by EDLines [57], and consecutive line segments are combined together to generate circular arcs. EDCircles detects arcs using three consecutive line segments that turn in same direction, where the angle between the line segments must be within a predetermined threshold. The low angle threshold and high angle threshold are determined as 6° and 60° respectively. The extracted arc is tried to fit a circle and if the root mean square error of the fitted circle does not exceed a certain threshold, it is added to the list of arcs. The detected initial arc is then extended by adding more line segments.

The next step generates candidate circles by joining arcs. After sorting of the arcs with respect of their length in descending order, the arcs that have close radii and center

are merged. The extended arc is tried to fit a circle. If the extended arc forms at least 50% of the circumference of the fitted circle, the circle is added to the candidate circle list. The remaining arcs are then used to generate ellipse candidates.

The last step of the algorithm is the validation of the candidate circle. This process is carried out with Helmholtz principle [46–49]. According to the Helmholtz principle, a geometric structure is perceptually meaningful, if the number of occurrences of the geometric structure is very small in a random situation. EDCircles adapts Helmholtz principle to validate circles and ellipses via redefining the number of false alarms (NFA) as in equation 3.1.

$$NFA(n,k) = N^5 \sum_{i=k}^{n} \binom{n}{i} \cdot p^i \cdot (1-p)^{n-i} \tag{3.1}$$

In equation 3.1, n is the length of the circle or ellipse, and k is the number of aligned pixels. The probability *p* is the accuracy of alignment between the level line angle of the pixel and the line segment. N^5 signifies the number of potential ellipses in an image with size $N \times N$. 5 degrees of N represents properties of ellipse, i.e., center coordinates of the ellipse, the major and minor axis, and its orientation. The candidate circle is valid if $NFA(n,k)$ of the circle is equal or smaller than a certain threshold epsilon (ϵ), which is fixed to 1, and corresponds to 1 false detections per image.

3.2. Curvature Aided Hough Transform For Circle Detection

Curvature aided circle detection (CACD) [54] presents a circle detection algorithm based on Hough transform benefiting from the curvature feature while estimating the radius and the center of the circle. Because Hough transform demands high memory and computation time, it needs prior information to speed up the running time and to decrease memory demand. CACD obtains this prior information about the radius and the center from the curvature analysis of the detected curves. Algorithm 3.1 presents the steps of the CACD.

CACD utilizes the well-known Canny edge detector to obtain the edges of a given image. Curves are determined by connectivity analysis. While doing connectivity analysis, the curves that are close to each other are merged, and thus, curves are generated. After curves are obtained, curvature calculation is performed for each curve to find the

Algorithm 3.1 *Algorithm of the curvature aided Hough transform*

Symbols used in the algorithm:
I: Input image
C: Validated circles

I. Preprocessing, image denoising, smoothing
II. Edge detection by Canny
III. Curve extraction
IV. Curvature radius estimation
for each layer **do**
  V. Accumulation with gradient direction and curvature heuristic
  VI. Detect circle center
  VII. Detect radius from radius histogram
  VIII. Save parameters of the candidate circle
end for
for each candidate circle **do**
  IX. If the circle has enough edge pixels, add to C.
end for
return C

radius. Calculation of the curvature is carried out by Gaussian smoothing for each point in the curve. The radius to be gathered from the curvature is calculated by the inverse of the curvature if the curvature is in range 2/3 of the maximum size of the image and 1. Otherwise, it will be 0 at that point. When the obtained radius from the curvature is not precise, it is indicated that it is not enough for circle detection. Circle center is determined by finding outlier points in accumulation. This means that if the point is at the center, it get more vote from the point on the circle using its estimated radius and gradient direction. After finding the circle center, the radius is taken to be the peak of the normalized radius histogram. A candidate circle is output only if it has enough edge points.

3.3. A Robust Circle Detection Algorithm Based On Top-Down Least-Square Fitting Analysis

Liu et al. [51] presents a robust, parameter-free circle detection algorithm based on top-down least-square fitting analysis (TDLSF) that makes use of an edge point chaining method. TDLSF has three main steps listed in algorithm 3.2. The first step is the edge segment extraction. Canny edge detector is carried out for extraction of the edge points. These edge points are then chained with a novel strategy. The strategy is based on moving direction history. The point to be added to the tail of the edge list is decided by examining

the history of moving direction rather than considering only the current point. Also, not only 3 neighbor points are searched but also 7 neighbor points are handled sequentially. After the edge map has been constructed, edge segments are divided into sub-segments to carry out top-down analysis and generate candidate circles. Division process benefits from history of moving direction. If the predicted edge point is not similar to the history of moving direction, the current edge segment is terminated and a new edge segment is initiated.

Algorithm 3.2 *Algorithm of the top-down least-square fitting analysis*

Symbols used in the algorithm:
I: Input image
C: Validated circles

I. Smoothing
II. Edge segment extraction
III. Candidate circle detection
IV. Circle validation
V. Add valid circles to C
return C

Circular arcs are extracted from divided edge segments with top-down analysis. According to the analysis, the curvature of sub-chains of the edge segment is inspected from the longest sub-chains to the shortest sub-chains progressively. After the arcs are detected, similar arcs are joined to generate candidate circles.

Validation process is similar to the one given in EDCircles. Two differences are indicated in the proposed work. First, instead of using N^5 in equation 3.1, $n \times (n-1) \times (n-2)/6$ has been used, where n is the number of edge points. The second difference is about the points to be used for validation. TDLSF performs the validation process in two subsequences, which are $\{p_1, p_3, \ldots, p_{2i-1}, \ldots\}$ and $\{p_2, p_4, \ldots, p_{2i}, \ldots\}$ for a candidate circle, rather than using all points on the sequence. The candidate circle is valid if it is validated with one of the two sequences.

3.4. A Fast And Robust Circle Detection Method Using Isosceles Triangles Sampling

Randomized circle detection methods randomly select three edge pixels and generate a circle using them. The fourth edge pixel is used for validation of the generated circle. However, it does not produce good results in noisy images. The work being examined in this section, a fast and robust circle detection method using isosceles triangles sampling named as ITCiD, makes use of isosceles triangles for circle detection by taking geometric properties of isosceles triangles into account [53]. The main idea of the work is to replace the sampling strategy of conventional randomized circle detection algorithms with the properties of isosceles triangles combined with gradient information. Algorithm 3.3 presents the general workflow of ITCiD.

Algorithm 3.3 *Algorithm of the isosceles triangles sampling*

Symbols used in the algorithm:
I: Input image
C: Validated circles

I. Gradient orientation estimation
II. Edge detection and contour extraction
III. Randomized sampling.
if finished **then**
  return C.
end if
if isosceles triangle criteria **then**
  if no **then**
    Go to III
  end if
end if
IV. Cluster analysis
if not matched **then**
  Go to III
end if
V. Refinement and verification
if no **then**
  Go to IV
end if
VI. Add detected circle to C, Go to III.

Isosceles triangle detection is carried out after gradient orientation estimation. The

idea is to look at the gradient directions of the two points, and the direction of the chord that connects these points. If the angles are similar, then it is assumed that these two points define an isosceles triangle. When the two edge points is very close or parallel to each other, then the angles would be similar. To prevent this situation, the dot product of the gradient vectors are taken, which should not exceed a certain threshold.

It is the fact that two points on the circle and the center make up for an isosceles triangle. The proposed method has utilized this fact to estimate the radius and the center of the circle. The center is estimated by intersecting the normalized gradient vectors of the edge pair supporting the isosceles triangle. After the center has been estimated, it is easy to find the radius from geometrical properties of the isosceles triangle. The edge pairs supporting the isosceles triangles are clustered according to their estimated radius and center for the sampling strategy. In the refinement step, circles are constructed with clustered edge pairs. Because of the noise in an image, centers of edge pairs fall into an area instead of a point. The refinement step then determines the center and the radius of the candidate circle.

4. OTCIRCLES: CIRCLE DETECTION FROM ORIENTATION TRANSFORM (OT) ARCS

Recently published circle detection algorithm, EDCircles [45], detects circles in the digital images with bottom-up strategy. This algorithm begins with extracting lines from the image afterwards arcs are generated. Then, they are combined into circles using several rules. EDCircles runs in real-time, namely that it is greatly fast, and produces good results. However, this algorithm fails in noisy images and complicated background. The main point in this failure is arc detection method of the algorithm. After EDCircles produces edge segments, it turns into small line segments. The arcs is produced using these small line segment. The problem in this process is that some arcs are missed due to the arbitrary turning directions of the line segments. Actually this situation causes to not detect existing circles in the image.

In this thesis, to overcome missing arc problem and make it more robust against to noise are benefited from recently published arc and line detection algorithm by Ding et al. [55], Orientation Transform (OT). OT may be simply defined as directly extracting the arcs instead of bottom-up strategy as in EDCircles from the edge segments. This method provides being developed algorithm for detection more existing arcs that cannot be detected EDCircles. Finding more existing arcs actually increases detection performance of

Algorithm 4.1 *Algorithm of OTLines*

Symbols used in the algorithm:
I: Input image
L: Line list
A: Arc list

for each chain **do**
  if satisfies the smallest eigenvalue criteria **then**
    I. add it to L
  else
    Orientation transform
    for decomposed part after orientation transfom **do**
      II. Detect line segments and circular arcs from smooth curve candidates
      III. Add the detected lines to L
      IV. Add the detected arcs to A
    end for
  end if
end for
return L and A

the algorithm.

In the following sections, orientation transform [55] is explained, and then clarified how OT arcs can be carried out in a more robust circle detector. In this thesis, proposed circle detection algorithm is named as OTCircles.

4.1. Orientation Transform (OT)

Recently, Ding et. al. [55] has proposed a novel line and arc detection algorithm named as OTLines, which simultaneously detects arcs and line segments from an image. First, OTLines extracts edge segments from the image using EDPF [50] and segments are divided at junction points. Junction point is determined when an edge point in an edge segment is neighbor to another edge segment. Then, divided edge segments are analyzed for straightness based on a principal component analysis. After this analysis, the edge segment that is not considered to be a straight line corresponds to discrete curve. Discrete curves are examined by orientation transform to detect corners and smooth curve candidates. The final step of the OTLines extracts arcs from the smooth curve candidates. The remaining continuous points are then returned as line segments. Algorithm 4.1 gives a pseudocode of OTLines. Orientation transform is the core step because it detects corners, lines, and arcs.

Orientation transform [55] maps whole points of the discrete curves to the orientation space. This space involves curve length, which is the amount of pixels in the edge segment, s and the tangent angle θ in horizontal and vertical axises respectively. Orientation transform holds discrete curve as horizontal lines and oblique lines. Thus, line or arc detection problem transforms into horizontal and oblique line detection problem in orientation space respectively. Orientation transform consists of three step:

1. Local shape description and orientation estimation
2. Orientation revision
3. Corner detection

First step is carried out with calculation of the eigenvector (v_{ik}) and eigenvalue (λ_{ik}) of the covariance matrix of the nearest neighborhood of the point p_i, which is the point on the discrete curve, using equation 4.1.

$$e = [v_{i1}, v_{i2}] = \begin{bmatrix} \frac{s_{12}}{\sqrt{(\lambda_1 - s_{11})^2 + s_{12}^2}} & \frac{s_{12}}{\sqrt{(\lambda_2 - s_{11})^2 + s_{12}^2}} \\ \frac{\lambda_1 - s_{11}}{\sqrt{(\lambda_1 - s_{11})^2 + s_{12}^2}} & \frac{\lambda_2 - s_{11}}{\sqrt{(\lambda_2 - s_{11})^2 + s_{12}^2}} \end{bmatrix}$$

$$\lambda_{ik} = \frac{1}{2}\left(s_{11} + s_{22} \pm \sqrt{(s_{11} - s_{22})^2 + 4s_{12}^2}\right), (k = 1, 2)(\lambda_{i1} > \lambda_{i2})$$

$$s_{11} = \frac{1}{L}\sum_{k=1}^{L}(x_k - x_m)^2$$

$$s_{22} = \frac{1}{L}\sum_{k=1}^{L}(y_k - y_m)^2$$

$$s_{12} = s_{21} = \frac{1}{L}\sum_{k=1}^{L}(x_k - x_m)(y_k - y_m) \tag{4.1}$$

where x_m and y_m are are the mean of the x and y coordinates of the points P_i. P_i is consisted of the nearest neighborhood of p_i within range $L = 2h + 1$. Here h is a certain small neighborhood size. In the paper of OT, h is set to 5 [55]. In this case, the range would be $L = 11$ pixels. When the smallest eigenvalue of P_i is smaller than certain threshold (τ), then P_i is considered to be a straight line and then the orientation of point p_i may be calculated carrying out equation 4.2.

The study of OTLines is inferred two results from the calculation according to [58]. First one is about straight lines. If the smallest eigenvalue is smaller than a certain threshold (κ), the set of points in P_i is straight lines. In this situation, the tangent orientation may calculate using equation 4.2. Second, if the smallest eigenvalue is higher than a certain threshold (κ), P_i may be considered as parabola or cubic parabola.

$$\theta_i = tan^{-1}\left(\frac{\lambda_{i1} - s_{11}}{s_{12}}\right), \theta_i \in (0, 180] \tag{4.2}$$

The next step in orientation transform is the orientation revision. The aim of this step is to maintain the continuity of orientation in the orientation space because of the orientation period caused by discontinuities at 180 degrees. The mentioned situation can be clearly in the Figure 4.1. When looked at first row, second column illustrates the orientation and next column its revised orientation. The curve begins at approximately 55 degrees. While walking on the circular arc, the angle raises to 180 degrees when reached top of the shape. After that point, the angle starts with 0 degree and again the angle reaches 180 degrees at about 365th pixel, then again turns back to 0 degree. To achieve continuity between these oblique lines, the angles are shifted 180 degrees until oblique

lines merge as far as possible. Also, as shown in third column, separated oblique lines in second column are merged into one oblique line starting from 55 degrees to 400 degrees. Other rows show similar operation for different shapes.

The last step, OTLines has indicated that multiplication of eigenvalues produces curvature of the point p_i. Curvatures of whole points in discrete curve are calculated with equation 4.3.

$$\kappa(p_i) = \begin{cases} \lambda_{i1}\lambda_{i2}, (\lambda_{i2} > \tau) \\ 0, \text{otherwise} \end{cases} \quad \textbf{(4.3)}$$

Here, $\kappa(p_i)$ represents curvature of i^{th} point. The detection of the corners, is carried out with the curvature analysis. Curvature of whole pixels on the discrete curve are calculated by equation 4.3. According to the equation, if the smallest eigenvalue of P_i is greater than τ, then the pixel p_i is assumed to be a corner candidate. Else, the curvature would assume to be zero. Thus, smaller values are suppressed for preventing much more local maximum pixels, which means having less corner candidates. The corner in the curve is determined with local maximum of candidate corners. Namely, the candidate corner which is local maximum due to the curvature will be corner. In the last step, the discrete curve is divided into several segments at the found corners.

Figure 4.1 demonstrates the five synthetic images and their orientations, revised orientations and corners in curvature form. When analyzed row by row, first row is completely circular shape and consists of several oblique lines; but after revision, it becomes one one oblique line. Thus, this edge segment is detected as one arc. The next column three oblique line and two horizontal line before revision. The segment starts somewhere in right-bottom of the arc. The movement passes from 90 degrees and reaches to 180 degrees. Then, it passes from 90 and 180 degree lines. Finally, it ends with remaining pixels in the arc. Its revised version simplifies the orientation with merging last two oblique line as shown in third column. The last column shows us three corner, which are local maximums of curvatures, at intersection of arc with 90 degree line, 90 degree line with 0 degree line, and 0 degree line with arc. In the second row, OT would produce two line and two arc segments. In the third row, segment starts at about intersection of two line segments. After revision, two separated oblique lines are merged into only one single oblique line. Because of the position of the beginning pixel, only two corner is detected in this shape. One of them is at intersection of horizontal line with arc and other is at

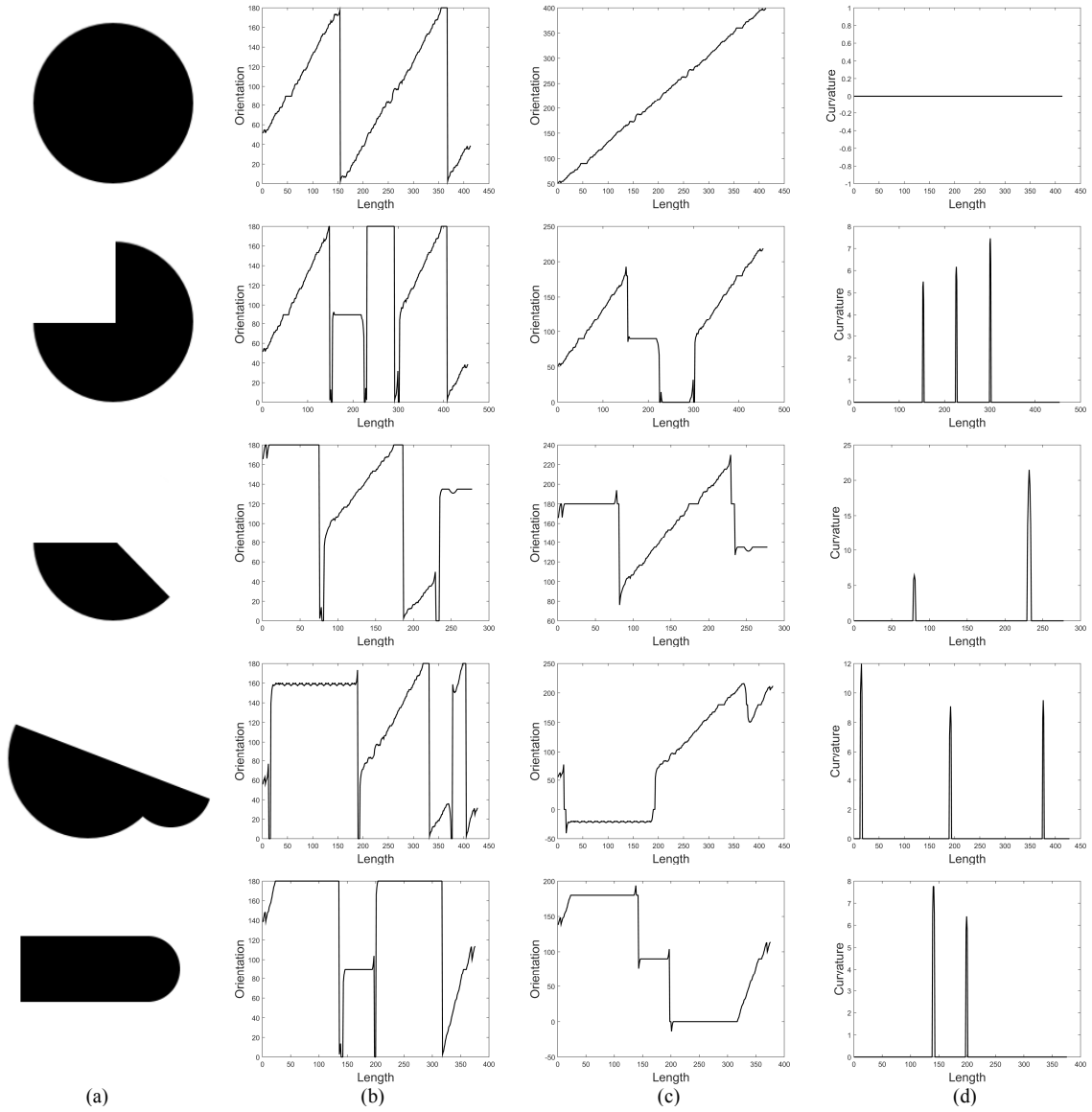


Figure 4.1. *(a) Some synthetic images, (b) Orientations before revision, (c) Orientations after revision, (d) Detected corners using the curvature feature.*

intersection arc with other line. In the next row, segment starts about intersection of long straight line segment with small arc segment. The movement is from small arc to long straight line. After revision, almost two oblique line for the two arc and one horizontal line is occurred. Three corners is detected successfully. In the last row, OT produces three horizontal lines and two oblique line; but it detects only two corners at intersection of straight lines. Furthermore, the corners at smooth transition from arc to straight line cannot be detected. However, orientation space is able to give us these corners since the shape is presented into oblique and horizontal lines.

4.2. OTCircles

OT directly extracts arc from the edge segments efficiently as mentioned previous section. Also, EDCircles [45] has good arc joining method and validation step with high precision. However, EDCircles fails while extracting arc when it does not find successive three line in same direction. The basic idea of OTCircles is to merge OT arcs with EDCircles to make a circle detector with high performance.

Algorithm 4.2 describes the basic steps of EDCircles [45]. EDCircles begins with extracting the segments of the given image using EDPF [50]. Then, it finds candidate circles from circular closed edge segments at the beginning. The remaining edge segments are turned into small line segments. Arcs are generated from combining the line segments as in steps II and III. EDCircles needs at least three successive line segments with in the same direction to generate an arc. Generated arcs are then turned into candidate circles in step IV. In steps V and VI, if a candidate circle successfully passes from the validation step, the circle is one of the output circles.

Algorithm 4.3 describes the basic stages of the developed circle detector, OTCircles,

Algorithm 4.2 *Circle Detection by EDCircles*

Symbols used in the algorithm:
I: Input image

EDCircles(I)
I. EdgeSegments = EDPF(I);
II. LineSegments = EDLines(EdgeSegments);
III. Arcs = ConvertLineSegments2Arcs(LineSegments);
IV. CandidateCircles = JoinArcs(Arcs);
V. FinalCircles = ValidateCircles(CandidateCircles);
VI. return FinalCircles;

Algorithm 4.3 *Circle Detection by OTCircles*

Symbols used in the algorithm:
I: Input image

OTCircles(I)
I. EdgeSegments = EDPF(I);
II. Arcs = OT(EdgeSegments);
III. CandidateCircles = JoinArcs(Arcs);
IV. FinalCircles = ValidateCircles(CandidateCircles);
V. return FinalCircles;

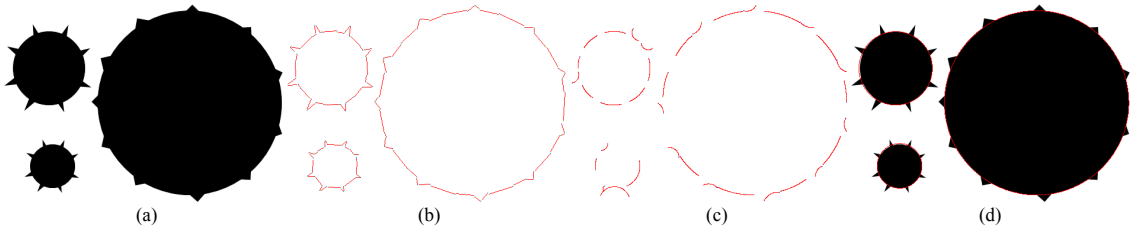


Figure 4.2. *(a) A synthetic image with three circles, (b) Line segments by EDLines, (c) Arcs detected by OT, (d) 3 circles detected by OTCircles.*

in this thesis. The differences between two algorithm is that arc detection method of EDCircles in steps II and III is swapped with the arcs produced by orientation transform. Other methods of the EDCircles do not changed. Shortly, orientation transform finds out arcs from the edge segments in a single step instead of using several steps of arc detection method of EDCircles so that it has more robust arc detection performance. Thus, produced better arcs would increase circle detection performance as being mentioned in following section.

Efficiency of the orientation transform arcs is demonstrated in Figure 4.2. According to the given example, it has three synthetic circular images in different sizes. The point to be taken consideration is that OTCircles detects all circles in the image while EDCircles could not do it for anyone. The reason why EDCircles could not detect any circle is that EDCircles needs at least three sequential lines in same turning direction to generate arc. Figure 4.2(b) shows the lines extracted by EDLines [57], which they will be used for arc generation. Each arcs in the image has two successive line; but it is not enough for EDCircles to generate an arc. Figure 4.2(c) is the output of the orientation transform arcs and it nearly extracts all arcs from the image. In OTCircles algorithm, the arcs produced by orientation transform are joined to generate candidate circles as seen in step III. Thus, in Figure 4.2(d) shows the validated circles, which are candidate circles passing the successfully from the validation step and drawn over the original image with red color.

Orientation transform enables to detect most of the arcs particularly in images with noisy and complicated backgrounds. Because orientation transform extracts the circular arcs from the given edge segment without any intermediate step as in EDCircles, it works well in the edge segment gained from the noisy images and extract arcs better than EDCircles. Consequently, OTCircles detects more arcs and then more circles considering to the EDCircles.

5. QUANTITATIVE PERFORMANCE EVALUATION METHODOLOGY

Evaluation methodology of the performance of a circle detector is mentioned in this section. Quantitatively evaluation of circle detection algorithms is carried out using human annotated circles with their ground truth. In literature, evaluation process of the proposed algorithms is carried out with a limited number of images without ground truth. This situation prevents to objectively and quantitatively compare the relative performance of different circle detection algorithms. To solve this problem, a rich dataset for evaluation of circle detection algorithms has been collected as mentioned in chapter 2.

In the following sections, the performance evaluation methodology of the circle detection algorithms is explained. Firstly, how to perform pairwise circle-circle matching between a ground truth circle and being detected circles how to obtained by an algorithm are explained . Then, how to evaluate the overall performance of a set of ground truth circles with the set of detected circles is described.

5.1. Pairwise Circle-Circle Matching

For a circle set produced by a circle detector for an image and the image's ground truth circles, which are human-annotated circles, should be matched for the precision-recall-Fscore metrics for quantitative evaluation. The intersection of the sets detected circles, D, and ground truth circles, GT, should be computed to get precision and recall. Circle-circle matching between ground truth and algorithm results is utilized with the best match criterion. The matching process is evaluated with overlap ratio between GT and D in Fscore metric. To determine the best match pair, each circle in ground truth is matched with all detected circles and generates candidate pairs. Fscore of each candidate pairs is calculated with due to the areas of the ground truth circle, detected circle, and intersection of them. Intersection of circles [66] is calculated with carrying out equation 5.1.

$$\begin{aligned} A_{intersection} =& r^2 cos^{-1}\left(\frac{d^2+r^2-R^2}{2dr}\right) + R^2 cos^{-1}\left(\frac{d^2+R^2-r^2}{2dR}\right) \\ & -\frac{1}{2}\sqrt{(-d+r+R)(d+r-R)(d-r+R)(d+r+R)} \end{aligned} \tag{5.1}$$

Here, r and R are the radii of the circles, and d is the distance between two circles centers as illustrated in Figure 5.1. $Match\_score$ is calculated with harmonic mean of precision

Table 5.1. *Each row represents the best pair of matched circles in Figure 5.2.*

GT Circle (x, y, r)	Detected Circle (x, y, r)	GT Circle Area	Detected Circle Area	Overlap Area	Precision	Recall	Match-Score
(165, 155, 105)	(180, 155, 100)	34636.06	31415.93	29780.82	0.9480	0.8598	0.9017
(410, 155, 50)	(415, 155, 55)	7853.98	9503.32	7853.98	0.8264	1.0000	0.9050
(590, 155, 40)	(595, 155, 65)	5026.25	13273.23	5026.25	0.3787	1.0000	0.5494

and recall of intersection area using equation 5.2.

$$
\begin{aligned}
Precision &= \frac{A_{intersection}}{A_{detected}} \\
Recall &= \frac{A_{intersection}}{A_{GroundTruth}} \\
Match\_score &= \frac{2 \times Precision \times Recall}{Precision + Recall}
\end{aligned}
\tag{5.2}
$$

Candidate pairs are sorted by descending order according to their match score. Thus, the most similar pair would be top of the list. Beginning from top of the candidate list, each pair is checked whether overlap ratio passes 80% threshold. If the pair exceeds the certain threshold, the detected circle in the pair is marked as true positive (TP) and the circles in the pair is removed from ground truth and also result circle list. In this way, these circles will not be examined later since they are matched. The remain circles in the ground truth is false negative circles (FN) and the remain circles in result circle list is false positive circles (FP).

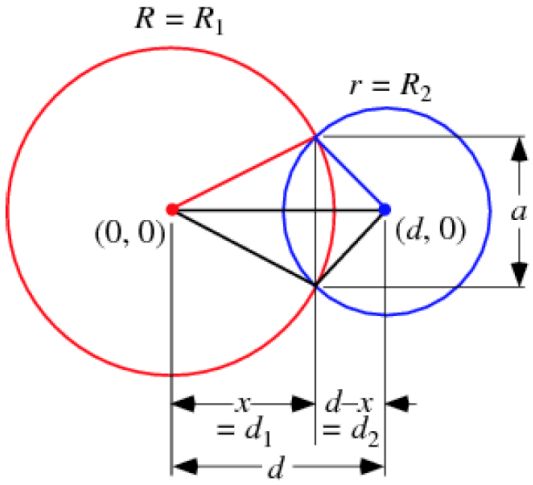


Figure 5.1. *Circle-circle intersection*

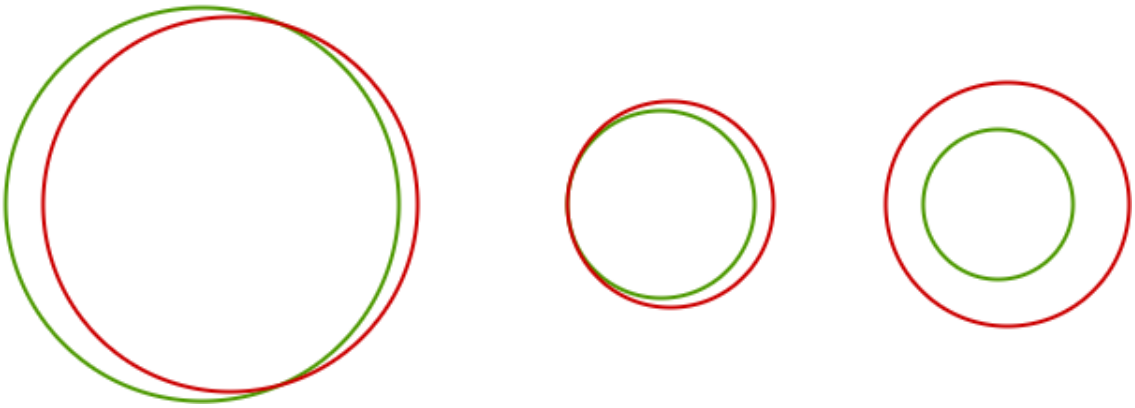


Figure 5.2. *Green circles are the ground truth (GT), and red circles are the detected circles (D).*

Figure 5.2 demonstrates an example for calculation of Fscores. Here, the green circles define the ground truth circles, and the red ones define the detected circles. Table 5.1 gives results of area calculations, precisions, recalls, and match scores of the best matching circle pairs. It may be seen from the table that match scores of the first and second pairs exceeds the certain threshold 0.80. Furthermore, these red circles may be considered as true positives. However, macth score of the last pair do not able to exceed certain threshold so that it cannot marked as a true-positive. Consequently, 2 true-positive, 1 false-positive, and 1 false-negative is produced for the example image.

5.2. Overall Performance Of The Circle Detection Algorithm

In this section, how to calculate overall performance of a circle detector on a dataset is described. Cumulative calculation of precision, recall and Fscore values for dataset needs the cumulative sum of TP, FP, and FN for overall images in dataset.

$$\begin{aligned} ImagePrecision &= \frac{TP}{TP+FP} \\ ImageRecall &= \frac{TP}{TP+FN} = \frac{TP}{GT} \\ ImageF\_score &= \frac{2 \times ImagePrecision \times ImageRecall}{ImagePrecision + ImageRecall} \end{aligned} \tag{5.3}$$

Circle – circle matching process is applied to all images for circle detector performance. First overall precision and recall of the circle detection algorithms is calculated running the dataset. Then, overall performance of the algorithms is determined in Fscore

metric, which is harmonic mean of the recall and precision using equation 5.3.

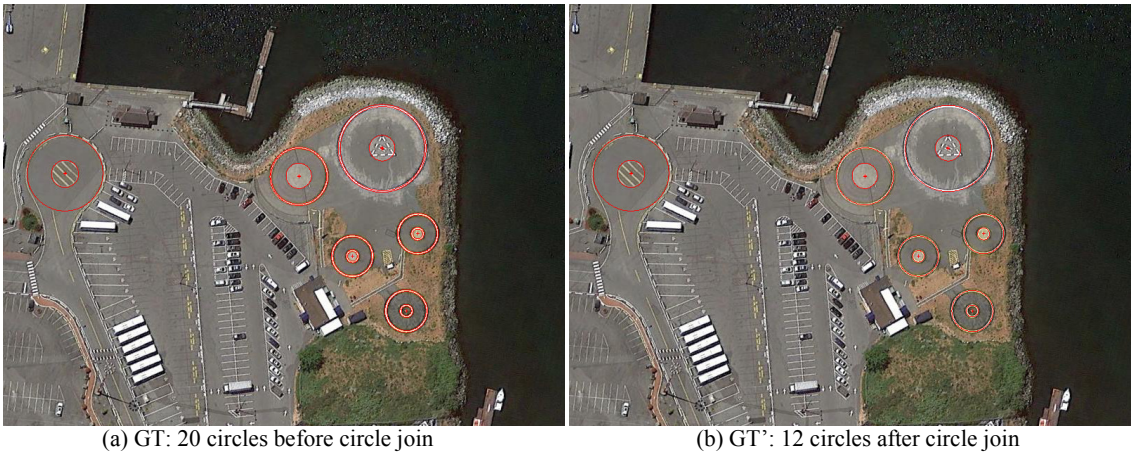
5.3. Circle Joining

Algorithms may detect all circles without consideration of closeness of the detected circles. Also, some algorithms do not detect circles overflowing from the image or small circles. The aim of the circle joining process is to normalize the results all circle detection algorithms and the ground truth circles based on some principles for a fair comparison. The criteria for circle join are listed below:

- A circle is discarded if its radius is less than 5 pixels
- A circle is discarded if more than half of circle's area exceeds the image
- Two concentric circles are joined if:
  - Center distance smaller than 15 pixels
  - Radius difference smaller than 4 pixels
  - Circle fit error of joined circle, which is generated with both circles' pixels, is less than 1.75 pixels

Both the ground truth circles and the detected circles by an algorithm are preprocessed using the rules given above, and the obtained circles are used for quantitative evaluation.

Figure 5.3 shows an example of concentric circles and joining of them. The image Satellite24 in AUCDB200, Figure 5.3, has 20 ground truth circles before joining pro-



(a) GT: 20 circles before circle join (b) GT': 12 circles after circle join

Figure 5.3. *Ground truth (GT) circles before and after circle join. Many of the concentric circles with small radius difference have been merged together.*

cess. Then, since the most of the concentric circles have small radius differences, they are merged into one circle for each one and ground truth circle count decreases to 12 circles. These decreased ground truth circles are carried out in quantitative performance evaluation.

6. EXPERIMENTAL RESULTS

6.1. Performance Comparison Of The Algorithms

The performance of proposed circle detection algorithm is quantitatively evaluated within the precision-recall-Fscore metrics using proposed dataset AUCDB200. The proposed algorithm's performance is compared with four recently published circle detection algorithms consisting of Top-Down Least Square Fitting Analysis (TDLSF) [51], Curvature Aided Circle Detection (CACD) [54], Isosceles Triangle Sampling (ITCiD) [53], and EDCircles [45].

Circle detection results for six images from each group of AUCDB200 are illustrated in Figure 6.1 for all algorithms which are handled in this thesis. First column presents human annotated ground truth. The following columns also present detection results of the algorithms for the concerned example images. OTCircles, which is in the last column, gains better detection results than others. when the results in the Figure 6.1 is analyzed in detail, the image in the first row includes 22 circular beads and EDCircles and OTCircles detects 15 of them. However, the other algorithms produced quite far results when compared with EDCircles and OTCircles. The next image includes 78 circles. OTCircles detects 68 circles, which is the highest result for the image, while the second best algorithms EDCircles and ITCiD only detect 56 circles of them. The third row in the figure is a satellite image. It includes 12 circles. OTCircles detects 11 circles correctly, which has one more circle than EDCircles. Also CACD detects 14 circles with 4 false positive circles. The next image has one big spherical object. This object is detected by OTCircles and CACD algorithms. The fifth image from the technical drawing group includes 5 circles. OTCircles could detect all of them; but EDCircles could find only 4 circles. Also, the others produces 2 false positive circles. The last image has 2 traffic signs. The circles in this image are detected by only OTCircles correctly. EDCircles could detect one of them. The others could not detect any of them. All of the circle detection results of the algorithms for the images in AUCDB200 dataset is available at OTCircles's Web site [64].

Precision, recall, and Fscore results of the concerned algorithms for AUCDB200 dataset are shown in Table 6.1. According to the table, OTCircles produces the best results with 0.9195 Fscore while CACD could do it with 0.6215 Fscore, which is the worst

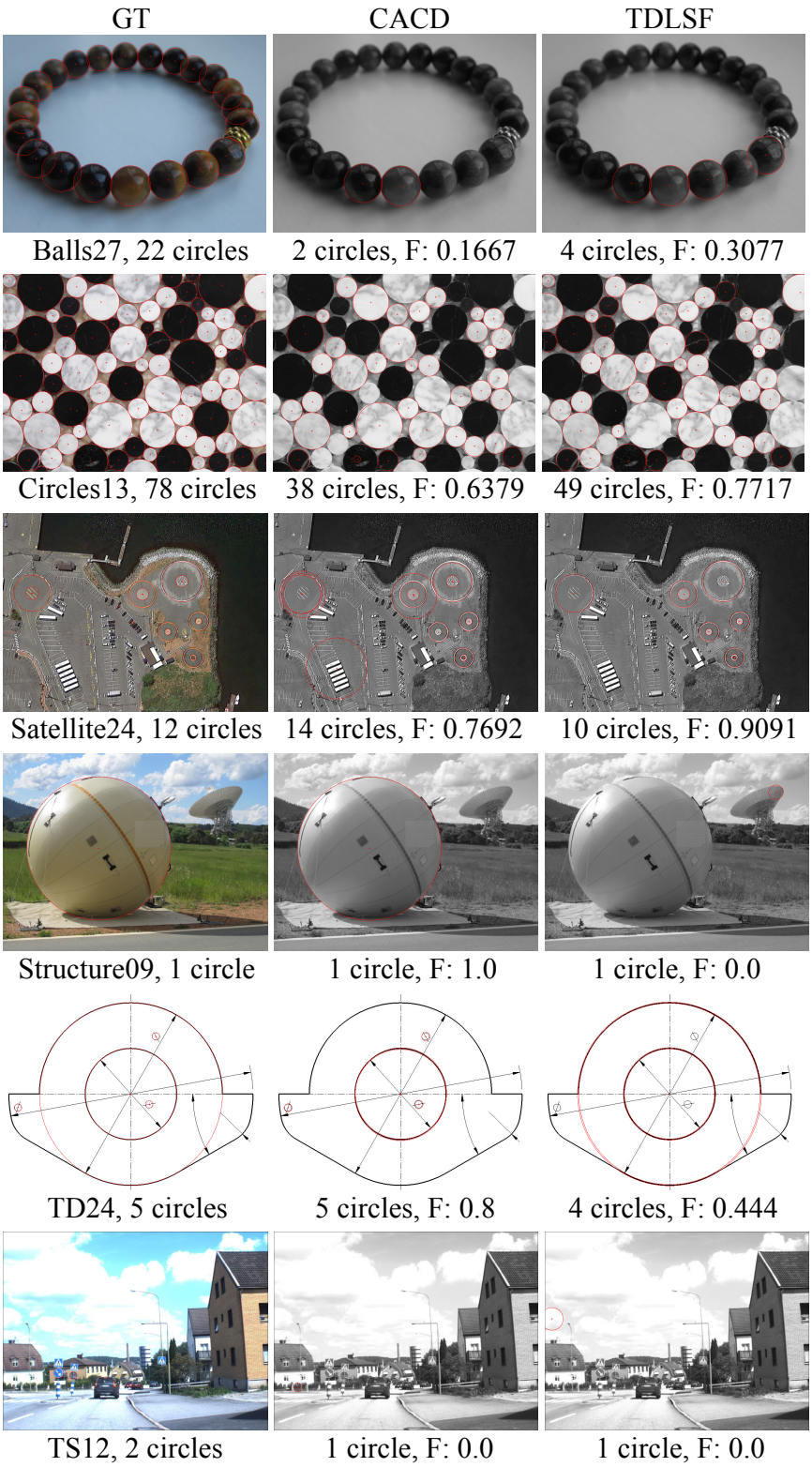


Figure 6.1. *Circle detection results for Curvature Aided Hough Transform (CACD) [54], Top Down Least Square Fitting (TDLSF) [51], Isosceles Triangle Sampling (ITCiD) [53], EDCircles, and OTCircles for 6 images, one from each category in AUCDB200. In all cases OTCircles produces the best results.*

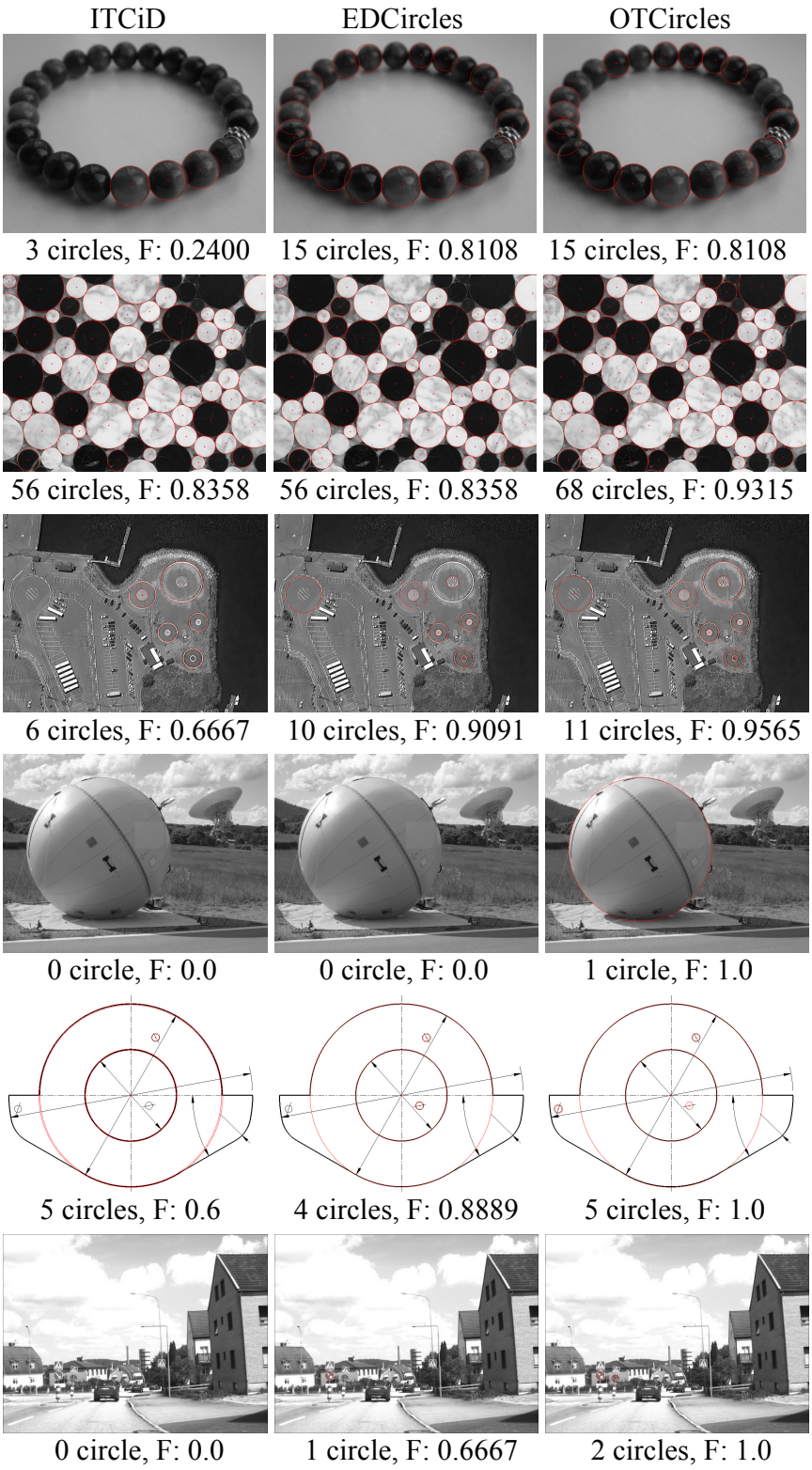


Figure 6.1. (Continued) *Circle detection results for Curvature Aided Hough Transform (CACD) [54], Top Down Least Square Fitting (TDLSF) [51], Isosceles Triangle Sampling (ITCiD) [53], EDCircles, and OTCircles for 6 images, one from each category in AUCDB200. In all cases OTCircles produces the best results.*

Table 6.1. *Overall precision, recall and Fscore values produced by the algorithms for AUCDB200. OTCircles outperforms all other detectors found in the literature.*

Algorithm	TP	FP	FN	Precision	Recall	Fscore
OTCircles	1679	74	220	0.9578	0.8841	**0.9195**
EDCircles	1520	38	379	0.9756	0.8004	0.8794
ITCiD	1260	32	639	0.9752	0.6635	0.7897
TDLSF	1253	199	646	0.8629	0.6598	0.7478
CACD	982	279	917	0.7787	0.5171	0.6215

Table 6.2. *Dissection of the Fscore of the circle detection algorithms for different categories in AUCDB200. OTCircles gives out the best performance both overall and for each category.*

Category	CACD	TDLSF	ITCiD	EDCircles	OTCircles
Balls	0.4160	0.6430	0.6899	0.8216	**0.8907**
Circles	0.7496	0.8411	0.8606	0.9331	**0.9556**
Satellite	0.8198	0.8362	0.8519	**0.9130**	0.8963
Structure	0.3333	0.6931	0.8200	0.7708	**0.9027**
TD	0.6240	0.7008	0.8415	0.8966	**0.9363**
TS	0.7374	0.7805	0.7363	0.8736	**0.8835**
Overall	0.6215	0.7478	0.7897	0.8794	**0.9195**

result among the algorithms. TDLSF and ITCiD algorithm results is quite far OTCircles since they do not have recall values as good as their precision values. The reason of low recall value is that they miss some of the existing circles. When EDCircles is analyzed, it can be seen that it has high Fscore value with 0.8794. However, it also misses more existing circles than OTCircles. OTCircles detects more circle than others and this provides increasing in the recall rate. Increasing of the recall rate causes a bit decrease in precision value. However, this decrease is greatly tolerated with increasing in recall value. Consequently, increasing in recall value of the OTCircles provides to get better Fscore value.

Table 6.2 separately shows Fscores of the algorithms for each groups in AUCDB200 dataset. Due to the results in the table, OTCircles yields the best results for the most of the groups. Only in satellite group, EDCircles beats the OTCircles with a little difference. It can be inferred from the results in the table that OTCircles could have opportunity to apply in various application areas.

6.2. Robustness Analysis Against Gaussian Noise

In this section, robustness of the algorithms is measured against the Gaussian noise. Gaussian noise with different levels are added to the images in the AUCDB200 dataset. Then, performance of the algorithms is evaluated for each level. Figure 6.2 shows an example over an image in the dataset. The noise is increased from $\sigma = 10$ to $\sigma = 50$ by tens. OTCircles keeps going to detect circles under high level of Gaussian noise. The other algorithms could not find any circle when sigma reaches 50.

Fscores of the algorithms in different Gaussian noise levels is plotted using the data in Table 6.3 in Figure 6.3. OTCircles keeps going to yield better performance under high level of Gaussian noise. When the noise level reaches $\sigma = 50$, OTCircles has 0.7803 Fscore value and it is still higher than other algorithms. Also, EDCircles has

Figure 6.2. *The performance of the algorithms as the added Gaussian noise is increased up to $\sigma = 50$ for a sample image in AUCDB200. OTCircles continues to show good performance even under high added noise while the performance of the other algorithms rapidly deteriorate.*

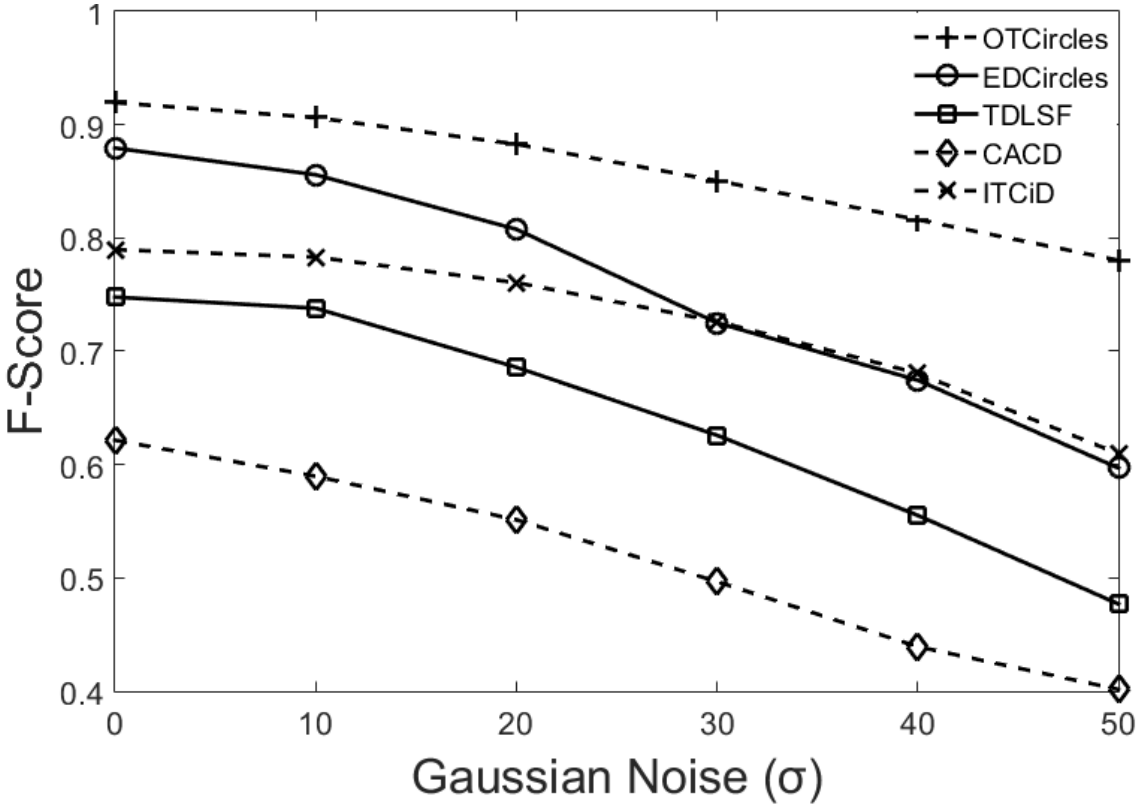


Figure 6.3. *Overall Fscores of the algorithms as the Gaussian noise increases up to* $\sigma = 50$. *OTCircles continues to perform fairly even under high added noise.*

dramatically falling under high Gaussian noise while passing from $\sigma = 20$ to $\sigma = 30$. However, OTCircles keeps going its stability under growing level of Gaussian noise. In conclusion, OTCircles is more robust than other algorithms.

Table 6.3. *Overall Fscores of the algorithms as the Gaussian noise increases up to* $\sigma = 50$. *Notice that OTCircles continues to perform fairly even under high added noise.*

Algorithm	**Gaussian Noise Level (sigma)**					
	0	**10**	**20**	**30**	**40**	**50**
OTCircles	0.9195	0.9062	0.8828	0.8505	0.8162	0.7803
EDCircles	0.8794	0.8558	0.8077	0.7251	0.6745	0.5970
ITCiD	0.7897	0.7831	0.7606	0.7264	0.6806	0.6096
TDLSF	0.7478	0.7380	0.6859	0.6259	0.5552	0.4770
CACD	0.6215	0.5897	0.5513	0.4967	0.4393	0.4016

6.3. Running Time Comparison

In this section, the running time of concerning algorithms are compared because the running time is a major factor for real-time applications. Table 6.4 demonstrates the average running time of the concerning algorithms for AUCDB200. The running times have been gained in a PC which has an Intel 3.4 GHz Core i7-4770 CPU property. The table infers that OTCircles is a bit slower than EDCircles. However, it is faster than other algorithms, which are ITCiD, TDLSF, and CACD. Results of the high detection performance has been extra running time cost for OTCircles, but it may be still considered as a fast algorithm.

Table 6.4. *Average running of the algorithms for AUCDB200. The running times were obtained in a PC with an Intel 3.4 GHz Core i7-4770 CPU.*

Algorithm	**Average Running Time (ms)**
OTCircles	30
EDCircles	24
ITCiD	41
TDLSF	310
CACD	8000

7. CONLUSIONS AND FUTURE WORK

Circle is a widely-encountered regular shape in difference places and for different tasks in daily life. In traffic, industry, eye tracking, biology and many situations. What it makes important is that circle involves meaningful and useful information in applied area. Because of its importance, several works have been proposed as mentioned in chapter 1.

In this thesis, two important works have been contributed to the literature. The first one is the circle detection dataset. The dataset consists of 200 images with size 800x600 pixels under six groups from several application areas. Also, the dataset has human annotation for each image being carried out as ground truth. This dataset is named AUCDB200 and used for quantitative evaluation of circle detection algorithms. Also, how to determine performance of a circle detection algorithm is explained using the human annotated dataset.

The second important work is a new circle detection algorithm in this thesis. The new circle detector benefits from orientation transform in recently proposed paper [55] and EDCircles [45]. The algorithm is named as OTCircles. The arcs are extracted by orientation transform. These arcs are merged to generate candidate circles with a arc joining algorithms as done in EDCircles. Finally, candidate circles which pass from the validation step are the output of the algorithm. The proposed circle detector, OTCircles, have been quantitatively compared with some circle detection algorithms in the literature, which are recently proposed algorithms consisting of CACD [54], TDLSF [51], ITCiD [53], and EDCircles [45]. The experiment results show that proposed circle detection algorithm, OTCircles, has the best overall Fscore with 0.9195 on the proposed dataset AUCDB200. Furthermore, OTCircles have produced better results than others on the most of the groups in dataset. It may be inferred from this result that OTCircles may apply to the different application areas. Also, robustness of the algorithms have been analyzed and it has been seen that OTCircles is more robust to the Gaussian noise than other algorithms. Whole materials including AUCDB200 dataset, the annotation and evaluation tools, human annotations, and the circle detection results of concerned algorithms is available at OTCircles's Web site [64].

The future work is to extend the proposed dataset so that it has more images in each category, and covers images from other applications areas. Also, looking into the problem of arc detection and trying to come up with our own arc detector that may result in better arcs to improve the circle detection rate are planed in future work.

REFERENCES

[1] Gonzales R, Woods R. (1992). *Digital Image Processing*. New York: Addison-Wesley.

[2] Forsyth D.A. (2002). *Computer Vision: A Modern Approach*. New Jersey: Prentice-Hall.

[3] Hiley, J. B., Redekopp, A. H., & Fazel-Rezai, R. (2006). A low cost human computer interface based on eye tracking. *Proceeding of Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE* pp. 3226-3229.

[4] Chen, Z., Yang, B., & Yin, F. (2015). An eye location based head posture recognition method and its application in mouse operation. *KSII Transactions on Internet and Information Systems, 9(3),* pp. 1087-1104.

[5] Bowyer, K. W., Hollingsworth, K., & Flynn, P. J. (2008). Image understanding for iris biometrics: A survey. *Computer Vision and Image Understanding, 110(2)*, pp. 281-307.

[6] Scholz, S., Mueller, T., Plasch, M., Limbeck, H., Adamietz, R., Iseringhausen, T., Kimmig D., Dickerhof M., & Woegerer, C. (2016). A modular flexible scalable and reconfigurable system for manufacturing of microsystems based on additive manufacturing and e-printing. *Robotics and Computer-Integrated Manufacturing, 40,* pp. 14-23.

[7] Davies, E. R. (2000). *Image processing for the food industry* (Vol. 37). Singapore: World Scientific.

[8] Mazzeo, P. L., Spagnolo P., Leo M., De Marco, T., & Distante, C. (2015). Ball detection in soccer images using isophote's curvature and discriminative features. *Pattern Analysis and Applications*, pp. 1-10.

[9] Arlicot, A., Soheilian, B., & Paparoditis, N. (2009). Circular road sign extraction from street level images using colour, shape and texture databases maps. *Workshop Laserscanning,* pp. 205-210.

[10] Fleyeh, H., & Davami, E. (2011). Eigen-based traffic sign recognition. *IET Intelligent Transport Systems, 5(3),* pp. 190-196.

[11] Berkaya, S. K., Gunduz, H., Ozsen, O., Akinlar, C., & Gunal, S. (2016). On circular traffic sign detection and recognition. *Expert Systems with Applications, 48,* pp. 67-75.

[12] Wang, G., Ren, G., Jiang, L., & Quan, T. (2014). Hole-based traffic sign detection method for traffic signs with red rim. *The Visual Computer, 30(5),* pp. 539-551.

[13] Bewes, J. M., Suchowerska, N., & McKenzie, D. R. (2008). Automated cell colony counting and analysis using the circular Hough image transform algorithm (CHiTA). *Physics in Medicine and Biology, 53(21),* pp. 5991-6008.

[14] Benligiray, B., Cakir, H. I., Topal, C., & Akinlar, C. (2015). Counting Turkish coins with a calibrated camera. *Proceeding of Image Analysis and Processing—ICIAP 2015*, pp. 216-226, Springer International Publishing.

[15] Hough, P. V. (1962). Method and means for recognizing complex patterns. *U.S. Patent 3069654*.

[16] Mukhopadhyay, P., & Chaudhuri, B. B. (2015). A survey of Hough transform. *Pattern Recognition, 48(3),* pp. 993-1010.

[17] Kierkegaard, P. (1992). A method for detection of circular arcs based on the Hough transform. *Machine Vision and Applications, 5(4),* pp. 249-263.

[18] Foresti, G. L., Regazzoni, C. S., & Vernazza, G. (1995). Circular arc extraction by direct clustering in a 3D Hough parameter space. *Signal Processing, 41(2),* pp. 203-224.

[19] Duda, R.O., Hart, P.E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM, 15(1),* pp. 11-15.

[20] Muammar, H., & Nixon, M. (1989). Approaches to extending the Hough transform. *Proceeding of Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on IEEE,* pp. 1556-1559.

[21] Stephens, R. S. (1991). Probabilistic approach to the Hough transform. *Image and Vision Computing, 9(1),* pp. 66-71.

[22] Kälviäinen, H., Hirvonen, P., Xu, L., & Oja, E. (1995). Probabilistic and non-probabilistic Hough transforms: Overview and comparisons. *Image and Vision Computing, 13(4),* pp. 239-252.

[23] Shaked, D., Yaron, O., & Kiryati, N. (1996). Deriving stopping rules for the probabilistic Hough transform by sequential analysis. *Computer Vision and Image Understanding, 63(3),* pp. 512-526.

[24] Galamhos, C., Matas, J., & Kittler, J. (1999). Progressive probabilistic Hough transform for line detection. *Proceeding of Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1, pp. 554-560

[25] Galambos, C., Kittler, J., & Matas, J. (2001). Gradient based progressive probabilistic Hough transform. *IEE Proceedings - Vision, Image and Signal Processing, 148(3),* pp. 158-165.

[26] Han, J. H., Kóczy, L., & Poston, T. (1994). Fuzzy hough transform. *Pattern Recognition Letters, 15(7),* pp. 649-658.

[27] Xu, L., Oja, E., & Kultanen, P. (1990). A new curve detection method: Randomized Hough transform (RHT). *Pattern Recognition Letters, 11(5),* pp. 331-338.

[28] Xu, L., & Oja, E. (1993). Randomized Hough transform (RHT): Basic mechanisms, algorithms, and computational complexities. *CVGIP: Image Understanding, 57(2),* pp. 131-154.

[29] Ji, Q., Xie, Y. (2003). Randomised Hough transform with error propagation for line and circle detection. *Pattern Analysis & Applications, 6(1),* pp. 55-64.

[30] Guo, S. Y., Zhang, X. F., & Zhang, F. (2006, August). Adaptive randomized Hough transform for circle detection using moving window. *Proceeding of Machine Learning and Cybernetics, 2006 International Conference on IEEE*, pp. 3880-3885.

[31] Jiang, L. (2012). Efficient randomized Hough transform for circle detection using novel probability sampling and feature points. *Optik-International Journal for Light and Electron Optics, 123(20),* pp. 1834-1840.

[32] Chen, T. C., & Chung, K. L. (2001). An efficient randomized algorithm for detecting circles. *Computer Vision and Image Understanding, 83(2),* pp. 172-191.

[33] Chung, K. L., & Huang, Y. H. (2007). Speed up the computation of randomized algorithms for detecting lines, circles, and ellipses using novel tuning-and LUT-based voting platform. *Applied Mathematics and Computation, 190(1),* pp. 132-149.

[34] Chung, K. L., Huang, Y. H., Shen, S. M., Krylov, A. S., Yurin, D. V., & Semeikina, E. V. (2012). Efficient sampling strategy and refinement strategy for randomized circle detection. *Pattern Recognition, 45(1),* pp. 252-263.

[35] Yao, J., Kharma, N., & Grogono, P. (2004). Fast robust GA-based ellipse detection. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on IEEE, 2,* pp. 859-862.

[36] Ayala-Ramirez, V., Garcia-Capulin, C. H., Perez-Garcia, A., & Sanchez-Yanez, R. E. (2006). Circle detection on images using genetic algorithms. *Pattern Recognition Letters, 27(6),* pp. 652-657.

[37] Das, S., Dasgupta, S., Biswas, A., & Abraham, A. (2008). Automatic circle detection on images with annealed differential evolution. *Hybrid Intelligent Systems, 2008. HIS'08. Proceedings of the Eighth International Conference on IEEE*, pp. 684-689.

[38] Dasgupta, S., Das, S., Biswas, A., & Abraham, A. (2010). Automatic circle detection on digital images with an adaptive bacterial foraging algorithm. *Soft Computing, 14(11),* pp. 1151-1164.

[39] Cuevas, E., Zaldivar, D., Pérez-Cisneros, M., & Ramírez-Ortegón, M. (2011). Circle detection using discrete differential evolution optimization. *Pattern Analysis and Applications, 14(1),* pp. 93-107.

[40] Cuevas, E., Osuna-Enciso, V., Wario, F., Zaldívar, D., & Pérez-Cisneros, M. (2012). Automatic multiple circle detection based on artificial immune systems. *Expert Systems with Applications, 39(1),* pp. 713-722.

[41] Cuevas, E., Ortega-Sánchez, N., Zaldivar, D., & Pérez-Cisneros, M. (2012). Circle detection by harmony search optimization. *Journal of Intelligent & Robotic Systems, 66(3),* pp. 359-376.

[42] Zhang, S. C., & Liu, Z. Q. (2005). A robust, real-time ellipse detector. *Pattern Recognition, 38(2),* pp. 273-287.

[43] Frosio, I., & Borghese, N. A. (2008). Real-time accurate circle fitting with occlusions. *Pattern Recognition, 41(3),* pp. 1041-1055.

[44] Wu, J., Li, J., Xiao, C., Tan, F., & Gu, C. (2008). Real-time robust algorithm for circle object detection. *Young Computer Scientists, 2008. ICYCS 2008. Proceeding of the 9th International Conference for IEEE,* pp. 1722-1727.

[45] Akinlar, C., & Topal, C. (2013). EDCircles: A real-time circle detector with a false detection control. *Pattern Recognition, 46(3),* pp. 725-740.

[46] Desolneux, A., Moisan, L., & Morel, J. M. (2000). Meaningful alignments. *International Journal of Computer Vision, 40(1),* pp. 7-23.

[47] Desolneux, A., Moisan, L., & Morel, J. M. (2001). *Edge detection by Helmholtz principle. Journal of Mathematical Imaging and Vision, 14(3)*, pp. 271-284.

[48] Desolneux, A., Moisan, L., & Morel, J. M. (2004). Gestalt theory and computer vision. *Seeing, Thinking and Knowing*, pp. 71-101.

[49] Desolneux, A., Moisan, L., & Morel, J. M. (2007). *From gestalt theory to image analysis: a probabilistic approach.* (Vol. 34). New York: Springer Science & Business Media.

[50] Akinlar, C., & Topal, C. (2012). EDPF: A real-time parameter-free edge segment detector with a false detection control. *International Journal of Pattern Recognition and Artificial Intelligence, 26(01),* 1255002.

[51] Liu, D., Wang, Y., Tang, Z., & Lu, X. (2014). A robust circle detection algorithm based on top-down least-square fitting analysis. *Computers & Electrical Engineering, 40(4),* pp. 1415-1428.

[52] De Marco, T., Cazzato, D., Leo, M., & Distante, C. (2015). Randomized circle detection with isophotes curvature analysis. *Pattern Recognition, 48(2),* 411-421.

[53] Zhang, H., Wiklund, K., & Andersson, M. (2016). A fast and robust circle detection method using isosceles triangles sampling. *Pattern Recognition, 54,* pp. 218-228.

[54] Yao, Z., & Yi, W. (2016). Curvature aided Hough transform for circle detection. *Expert Systems with Applications, 51,* 26-33.

[55] Ding, W., Wang, W., & Li, X. (2016). OTLines: A novel line-detection algorithm without the interference of smooth curves. *Pattern Recognition, 53,* pp. 238-258.

[56] Topal, C., Akınlar, C., & Genç, Y. (2010). Edge drawing: A heuristic approach to robust real-time edge detection. *Pattern Recognition (ICPR), 2010. Proceeding of the 20th International Conference on IEEE* pp. 2424-2427.

[57] Akinlar, C., & Topal, C. (2011). EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters, 32(13),* pp. 1633-1642.

[58] Lee, Y. S., Koo, H. S., & Jeong, C. S. (2006). A straight line detection using principal component analysis. *Pattern Recognition Letters, 27(14),* pp. 1744-1754.

[59] `http://dae.cse.lehigh.edu/DAE/?q=browse` Accessed: April, 2016

[60] `http://www.cs.cityu.edu.hk/~liuwy/ArcContest/ArcSegContest.htm` Accessed: April, 2016

[61] `http://benchmark.ini.rub.de/?section=gtsdb&subsection=dataset` Accessed: April, 2016

[62] `http://www.cvl.isy.liu.se/research/datasets/traffic-signs-dataset/download/` Accessed: April, 2016

[63] `GoogleEarth-https://www.google.com/earth/` Accessed: April, 2016

[64] OTCircles Web Site, `http://ceng.anadolu.edu.tr/CV/OTCircles` Accessed: June, 2016

[65] Circle fit algorithm, `http://www.dtcenter.org/met/users/docs/write_ups/circle_fit.pdf` Accessed: April, 2016

[66] Circle-circle intersection, `http://mathworld.wolfram.com/Circle-CircleIntersection.html` Accessed: April, 2016

CURRICULUM VITAE

Name-Surname : Gökhan ÇIPLAK

Birth Place and Year : AYDIN/Nazilli, 1990

E-mail : gciplak@anadolu.edu.tr

Education

- MS Degree in Computer Engineering Department
  Anadolu University, Eskişehir, Turkey November 2016
- Bachelor's Degree in Computer Engineering (English)
  Anadolu University, Eskişehir, Turkey 3.40/4.00 G.P.A June 2013
- Nazilli Anatolian High School ,Nazilli AYDIN
  Concentrationin English and Germany June 2008

Career

- Research Assistant, Anadolu University, Eskişehir, Turkey 2015 - ongoing
- BT Bilgi Teknolojileri 2013 - 2014

Papers Submitted to International Meetings

- Ciplak, G., & Telceken, S. (2015, November). Moving object tracking within surveillance video sequences based on EDContours. *Proceeding of 9th International Conference on Electrical and Electronics Engineering, ELECO*, pp. 720-723